DEVELOPMENT OF A SCHEDULING MODULE WITHIN AN INTEGRATED SOLUTION FOR THE EVALUATION OF PROCESS VARIANTS

Tim Neumann, Daniel Kretz, Joerg Militzer and Tobias Teich
University of Applied Sciences Zwickau, Dr.-Friedrichs-Ring 2a, 08058 Zwickau, Germany

Keywords: Proposal preparation, Scheduling, Genetic algorithm, Manufacturing, Process variant.

Abstract: The success or failure of small and medium sized enterprises (SME) is related to the handling of factors like individual customer demands, price pressure and the probability to deliver at the required date and time. Often such SME's are on the market for single-part or small-series production and want to be supplier for larger companies. Therefore, the decision makers of their customers have to investigate potential suppliers due to these mostly interrelated criteria. To increase these known factors during the proposal preparation is one possibility to enhance the market position of the SME. Thereby, a consideration of different variants of manufacturing a product and the premature investigation of resources and their capacities is necessary. Within the scope of this paper is introducing a conceptional framework for the evaluation of different process variants to manufacture a product. Thereby, we are using genetic algorithms to optimize and evaluate process variants including the necessary resources and their capacitive use in an evaluated period. Additionally, we want to introduce our prototypical implementation.

1 INTRODUCTION

The process planning is a significant phase after the product design in product lifecycle for the success or failure of a small and medium sized enterprise (SME). Especially in the branch of single-part or small-series production in industrial engineering is the process planning responsible for the improvement of criteria like individual customer demands, price pressure and the probability to deliver at the required date and time. Therefore, it is necessary to think about these factors and their dependencies as soon as possible. One possibility to reach an enhancement is the consideration of different ways for manufacturing a product, the premature investigation of resources which possibly could be used and their capacity, during the proposal preparation.

Currently, discombobulated application landscapes and heterogenic systems are preventing an efficient, integrated and automated product development. Therefore, it is indispensable to use the potential of an integrated solution of Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Computer Aided Process Planning (CAPP) and Enterprise Resource Planning (ERP).

We are developing a integrated solution which consists of seven fundamental process steps, as illustrated in figure 1. First, there is a feature based CAD drawing of a requested part. Thereby, a feature is an object for the description of parts and their geometrical, functional and technological properties

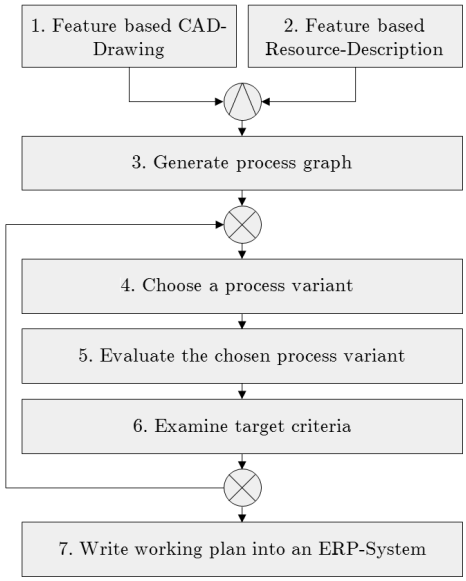

Figure 1: Structure of the parts.

Neumann T., Kretz D., Militzer J. and Teich T..
DEVELOPMENT OF A SCHEDULING MODULE WITHIN AN INTEGRATED SOLUTION FOR THE EVALUATION OF PROCESS VARIANTS.
DOI: 10.5220/0003531103930398
In *Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2011), pages 393-398
ISBN: 978-989-8425-75-1

(Ehrlenspiel et al., 2005). To realise this definition we utilise the STEP application protocol (AP) 224 from ISO 10303 (International Organization for Standardization, 2006). Next to the feature based CAD drawing, we implemented a feature based resource description which describes suitable machines and other resources as well as the features in the CAD drawing. This provides the ability to determine e.g. set-up times or accruing costs. We extract the master data for these resources from an ERP system.

With these two prerequisites we provide the possibility to generate a process graph in the third block. This graph includes all feasible variants for the production of the requested part and therefore consideres geometrical, functional and technological correctness. Thereby, it maps the expertise of a process planner to support the decision process and it is responsible for determining suitable manufacturing process variants which could be utilized for the production of the part or assembly.

After generating the process graph we have to choose a single process variant for the upcoming evaluation in the fourth block. Therefore we utilize genetic algorithms (GA) for scheduling and hence the evaluation of this process variant.

In the sixth step we examine the target criteria. If they are not met or not satisfying, we return to the fourth part and select another process variant. If all target criteria are met, or satisfying, we can continue with the process and submit the selected process variant as a working plan into the ERP system in a last step.

Within this paper we want to introduce the scheduling problem for the evaluation of a process variant. Afterwards we describe our genetic algorithm and its characteristics. After the theoretical concept we want to give a introduction to our software architecture of the whole integrated solution. This section is followed by our conclusion.

2 SCHEDULING PROBLEM

Amongst other things, placed orders and their delivery dates are different input information which we require for the scheduling process. These orders are not production orders, they are just placed orders we need to deliver. Additionally, we need information about available resources and their capacitive utilization. This information represents fixed production orders within an evaluated period. Finally we need information about the initially transferred process variant. This process variant consists of the parts list and the sequential working plan. The specification to sequential working plans based on the viewing at component level.

For this consideration we have defined a own scheduling problem that will be explained and classified as follows. For this reason we want to refer to the definition implemented by Graham et al. (Graham, R.L. ; Lawler, E.L. ; Lenstra, J.K. ; Rinnooy Kan, A.H.G., 1979) and T'kindt (T'kindt and Billaut, 2006).

Basically, n jobs $(J_j \mid j = 1,...,n;\ n \in \mathbb{N})$ should be produced on m machines $(M_i \mid i = 1,...,m;\ m \in \mathbb{N})$. Additionally is defined, that the number of operations O is divided into o_{jk} $(k \mid k = 1,...,l;\ l \in \mathbb{N})$ operations. This implies that o_{jk} is the k^{th} operation of job j. Therefore, it is defined, that every job J_j with $J_j \in J$ consists of a sequence of k operations (o_{jk}). Consequently is $o_{jk} \prec o_{j(k+1)}$. Hence, we define that the k^{th} operation of the job has to be executed before its successor.

For each step within the manufacturing process, a number of applicable machines $M_{jk} \subset M$ is assignable. Depending on the chosen machine M_i, each operation determines different production (execution) times p_{jki}. This is a derivation from the basic definition and describes the production time p_{jki} as time for the production of operation k of job j on machine i. We also define, for each machine distinct cost rates. We declare the cost rate for each time unit and machine as c_i.

Viewing at component level implies another deviation. The base definition defines a due date for each job as d_j. We have modified this definition to assign due dates only to jobs without any successor. Thus, we address the component at the top level of the part list. Accordingly, this associates the requested part.

Additionally, our approach supports the mapping of divergences and convergences in the production process. Figure 2 illustrates this aspect.

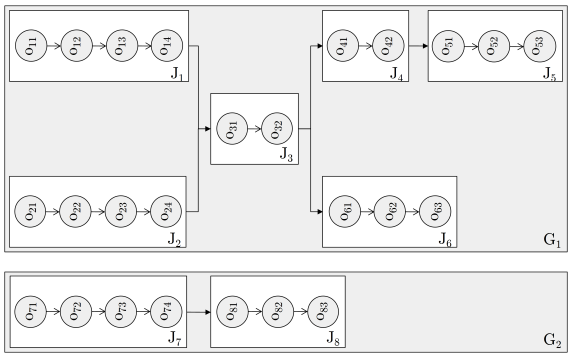

Figure 2: Example structure.

This example assume, that the jobs J_5 and J_6 are placed orders within the evaluated period. Accord-

ingly, the jobs J_1 to J_4 map the part list of these two products. Each level of the part list represent the sequential working plans. Finally we assume, that job J_8 represents the requested product.

The final aspect of our scheduling problem definition is the grouping of jobs. This approach results in a more balanced selection of jobs during the scheduling process. Hence, we define that jobs which are directly or indirectly related to other jobs belong to a common group.

After the definition of the scheduling problem, we want to classify the discussed problem. Therefore, we use the common $\alpha \mid \beta \mid \gamma$-classification.

Within this classification we can categorize all known scheduling problems. Thereby α describes the machine environment, β the job characteristics and γ the optimality criteria. For our previously introduced scheduling problem we define the following classification:

$$GMPM, m \mid prec, d_j, p_j \in [\underline{p_j}; \overline{p_j}], recrc \mid GP$$

The first parameter α describes the machine environment. Because of their restrictions, classifications types like *P* for identical parallel machines, *Q* (uniform parallel machines), *R* (heterogeneous parallel machines), *F* (Flow-Shop problems), *O* (Open-Shop problems) and finally *J* (Job-Shop problems) are not applicable for our problem definition. Certainly, the classification into Job-Shop problems is similar to our problem definition, but it does not fit exactly. In fact we only have an partial assignment between operations and machines. Therefore we need a generalized classification. Hence, we have identified our problem as a "General Shop Multi Purpose Machines Scheduling Problem" (*GMPM*). The second information in this field is *m* and is related to the amount of machines, which is unknown but fixed.

The second parameter β describes the job characteristics. To reflect the previously explained predecessor-successor relationships between jobs as well as operations, we use *prec* (precedence contraints). This definition is also very general and includes special cases. Next to the definition of the job structure, we have defined that d_j describes the information about due dates. Furthermore, we have defined the limits of production time $p_j \in [\underline{p_j}; \overline{p_j}]$.

The finally given information of our classification is *recrc* (recirculation). This indicates that a single job can be produced repeatedly on one machine.

The last parameter of the $\alpha \mid \beta \mid \gamma$-classification is the optimality criteria γ. As we mentioned initially, more than a single fact is responsible for the success or loss of a company. Consequently we require to utilize multi criteria optimisation. In fact, there are different multi criteria optimisation approaches possible (Loukil et al., 2005). We have selected the goal programming approach as introduced by T'kindt and Billaut (T'kindt and Billaut, 2006). This is reasoned by the fact that we can define main goals, like reducing selling prices by comparing accruing costs or reducing deviations from due dates by comparing completion times. This multi criteria optimisation will also be the basis for the fitness evaluation in the genetic algorithm.

3 GENETIC ALGORITHM

Genetic algorithms are often used for solving scheduling problems. The relatedness between the defined scheduling and the job-shop scheduling problem leads towards occupying with genetic algorithms (Syswerda, 1991).

The basics of genetic algorithms are developed by Holland (Holland, 1975) and De Jong (Jong, 1975). A illustration of the procedure of the genetic algorithm in generally gives (Vnyi, 2004).

First, we have to initialize a random population of solutions. We apply a serial genetic algorithm. Furthermore we use the generational replacement as described by (Holland, 1975). This means that one generation of solutions is completely replaced by their offspring. Additionally to the generational replacement we are using elitism to retain the best solutions.

The performance of a GA depends on an effective problem representation. Thereby, problem representation is an encoding of the actual optimization problem into structures of individuals. Within the encoding process we have to ensure that required information is completely mapped into an individual and avoid invalid solutions.

We have decided to develop a multi chromosome individual with operation-based representation (Gen and Cheng, 1997). To give an example, we have illustrated a multi chromosome structure in figure 3 in reference to our example graph.

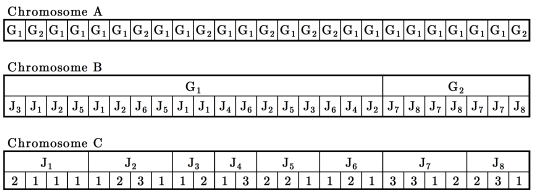

Figure 3: Example individual.

Thereby, chromosome A represents a permutation of groups. The Chromosome B consists of a permutation of jobs. Finally, with chromosome C we want to select a resource for each operation. As you can

see it is not a permutation of elements. Furthermore, we utilise an integer chromosome which represent an index of the selected machine or resource from a resource set. Every chromosome evolves independently from each other.

To compare the individuals we have to decode and calculate a fitness value. The decoding process of the individuals is an iterative process, beginning at the left side of the individual.

The first step is the selection of the first element of chromosome A. Referring to the illustrated example individual the first element of chromosome A is G_1. Therefore, we retrieve the first job from chromosome B, which is part of group G_1. Additionally this job has to be executable which means either there are no predeceeding jobs or they are all already executed completely. After the selection of the job we need to identify the next executable operation and an assigned machine from chromosome C. The index in the elements of chromosome c indicates the machine selection from a given set of assignable resources. Now, we have decoded one individual for the first time. We continue the decoding process for the next element in chromosome A in the same way. Consequently, we have to repeat this process for each element in chromosome A. After the last element we get a table with a complete planning sequence.

With such a table and the addition of machine capacity utilisation in the evaluated period we can develop a new machine utilisation plan. This is necessary because the pure processing time does not provide an evaluation of the solution quality.

On the one hand there is a solution with lesser total production time, but with a displacement of completion dates after the due dates. On the other hand, there could be a solution with an enormous increased total production time, but with a displacement of the completion times before the due dates.

Accordingly, for our algorithm we have selected the minimisation of the maximum weighted lateness L^{ω}_{max} which is an result of two different kinds of lateness, the earliness $E_j = \max\{0, d_j - C_j\}$ and the tardiness $T_j = \max\{0, C_j - d_j\}$. Thereby, C_j addresses the completion date and d_j the due date of a job. Furthermore it is possible that the minimisation of earliness is more important than the minimisation of tardiness. So we have defined the weighted lateness L^{ω}_{max}:

$$L^{\omega}_{max} = \sum_{j=1}^{n} (\omega_E \; E_j \; + \; \omega_T \; T_j)$$

Additionally to this value we have to evaluate the minimisation of the maximum production costs $K_M(p_{jki} \; ; \; c_i)$. This is an attempt to multi criteria optimisation of goal programming. We derive due dates and accruing costs of our products and we can compare them to our completion dates and the prices which have to be achieved.

To improve our solution we have to search better solutions within the solution space. Thereby, genetic operators help to improve our individuals. First of all, we have to define and determine these basic operators like the selection, the recombination operator and finally the mutation operator.

Before the recombination of some individuals, we have to select parent individuals. This is done by the selection operator (Nissen, 1997). Currently our algorithm applies the tournament selection. Thereby we chose a number of individuals for a tournament. This number depends on the size of the population reduced by the number of elitists and the percentage of elements we want to select for the tournament. The fittest individual within the tournament group is selected as a parent. To determine the second parent, we repeat this approach. With the two selected parents we can continue with the recombination. This procedure shall be continued until we have selected the required number of parents.

Recombination as the second operator is deemed to be the central operator in the algorithm. Because of the different kinds of chromosomes in one individual we have to separate the individual for the recombination.

Chromosome A and B are permutation chromosomes. This chromosome type requires specialised crossover operators. In our case we use the partially mapped crossover (PMX) (Ting et al., 2010). This operator is characterised by the preservation of genetic sequences without destroying the permutation character. Accordingly, it is possible, that good sequences are achieved.

Chromosome C is an integer chromosome and therefore, we have to use another specialised crossover operator. There are two basic operators. First, there is the uniform crossover (Syswerda, 1989) and second the N-Point-Crossover (Gwiazda, 2006). At the moment the decision which of these two possible crossover operators we are implementing is not made.

In addition to the recombination we use mutation operators. These are useful to established a balance between exploitation and exploration (Weicker, 2007).

Before starting with the mutation we have to determine a global or overall mutation rate reveals whether a chromosome is mutated or not. Additionally to this we have to define the local mutation rate which determines the probability of applying mutation operators and influences the strength of mutation in the genetic algorithm.

Given the fact that there are two different chromosome types we also need specialised mutation operators e.g. to preserve the permutation in chromosome A and B. To permute A and B we randomly selected different procedures for mutation. Therefore, we have forced to implement four different methods like Position-Based mutation, Order-Based mutation, Scramble-Based mutation or the Swap mutation (Syswerda, 1991). This concept supports the idea of parallelism in the algorithm.

The integer chromosome C is mutated by the adjoining mutation (Michalewicz, 1999).

With this last operator we are at the beginning of the procedure. For the next generation of individuals this algorithm has to be repeat until the stopping criterion is reached.

In the following section we want to introduce a implementation of this algorithm in our integrated solution.

4 IMPLEMENTATION

The developed and introduced framework for the evaluation of process variants is part of an integrated system. For this system we have presented a process structure in the first section of this paper. Now we want to introduce our implementation of this structure. Therefore, we have drafted a five tier architecture which reflects the different abstraction layers of our final software solution

The lowest level is the persistence layer. This layer is responsible to keep all information for different application states permanently. This data is saved on storage devices within files or databases.

For the integration and collaboration of the particular modules we define the communication layer. This layer provides access to the permanent required and accruing data of the persistence layer. Additionally, it is basis for the exchange of logical program data during runtime. The communication layer provides also the ability to handle the other modules e.g. to invoke succeeding processing functions or to provide state information about executed calculations and their results.

We define the application layer in the third tier. In detail it consists of four particular modules that consequently represent the implemented components of the program and algorithms. In the AP 224 CAD module CAD drawings are modelled. This module is necessary to instantiate the feature based product model. The second one is the process variants module. Thereby, we develop a system, that generates different suitable process variants to manufacture the requested part. Thereby, it maps the expertise of a process planner to support the decision process. The process variants are extended by the resource module. This contains information about assignable resources and their abilities which provides the ability to determine e.g. set-up times or accruing costs. Finally on this layer there is the scheduling module which is discussed within this paper.

For the implementation of the genetic algorithm in the scheduling module we are utilizing an optimization framework called EvA2(Unversity of Tuebingen,). EvA2 offers a huge range of heuristics including evolutionary algorithms. The reason for using EvA is quite obvious. EvA includes a library for the GA procedure that we have already discussed in this paper. The graphical user interface (GUI) of EvA provides a solution for an easy parametrization of the genetic algorithm. Another advantage of this framework is the integration of a visualization of the own optimization procedure. To implement our defined problemwe have mapped our problem into an abstract scheme. After the problem definition we have adapted the problem representation for EvA2. In addition to the problem definition and representation we have implemented the fitness evaluation. EvA2 uses these definitions in the genetic algorithm. We only had to implement specialized operators like the partially mapped crossover. Consequently, we do not have to worry about the handling of the genetic algorithm. The required data is extracted from an ERP system, in our case SAP. For the data exchange we need an interface to send and receive master and transaction data to or from an external application. SAP offers a solution for the data exchange between the ERP system and the APO (Advanced Planner and Optimizer). This is called the SAP Core Interface (CIF). Additionally to the CIF we use the SAP Java Connector (SAP JCo). This is a middleware which supports the development of SAP connected Java applications including incoming and outgoing ABAP calls. These ABAP calls are received and sent by a Java server and processed e.g. in the scheduling module. Therefore, we had to find, interpret and simulate all required outgoing and incoming function calls to use the containing data in the external scheduling module. Furthermore, we have to run the function calls in the SAP system to load the results from the scheduling module into the ERP system.

Therefore, the fourth layer is the control layer which avoids the isolation of processes which are running within our application modules. Additionally, this layer controls and coordinates the entire planning process and states of the particular modules.

The fifth layer is the presentation layer which rep-

resents the user interface for the interaction between users, especially designer, as well as mechanical engineers, and the application modules.

5 CONCLUSIONS

Within the scope of this paper we have introduced a concept for a scheduling module to evaluate process variants in the proposal preparation using an genetic algorithm. Therefore we have explained our problem definition and classification $\alpha \mid \beta \mid \gamma$ - classification and its rules for the scheduling problem. Thereby, we derived the using of a very general classification and we did not exclude any specialised definition.

Afterwards, we discussed the genetic algorithm to schedule a process variant and the workload in a company within a defined evaluation period. For this we use the widespread problem representation of the operation based representation. In a first version we developed a sequential genetic algorithm. With a more complex problem definition it is possible or necessary, that we are forced to use other structures like a parallel genetic algorithm. Within this definition we also introduced the the genetic operators selection, recombination and mutation.

The last section explained our software architecture and the implementation for the discussed procedure within a scheduling module. This includes the description of the five tier architecture and especially for the scheduling module the EvA2 framework. Additionally we have explained the data exchange between the ERP system and the developed application.

Future work includes tests of the algorithm, its configuration and its efficiency. Therefore we have to use established benchmarks or we have to develop a own benchmark. This should include different scenarios for a wide range of different cases. Furthermore, we have to develop an extended version of the optimisation program. This version should include the test results and changes of the algorithm configuration.

REFERENCES

Ehrlenspiel, K., A.Kiewert, and Lindemann, U. (2005). *Kostengnstig entwickeln und konstruieren - Kostenmanagement bei der integrierten Produktentwicklung*. Springer, Berlin, 5th edition.

Gen, M. and Cheng, R. (1997). *Genetic algorithms and engineering design*. Wiley series in engineering design and automation. Wiley & Sons Inc.

Graham, R.L. ; Lawler, E.L. ; Lenstra, J.K. ; Rinnooy Kan, A.H.G. (1979). *Optimization and approximation in deterministic sequencing and scheduling. A survay.* Annals of Discrete Mathematics 5. Gabler.

Gwiazda, T. D. (2006). *Genetic Algorithms Reference - Crossover for single-objective numerical optimization problems*, volume Volume I. Thomas Gwiazda.

Holland, J. (1975). *Adaption in natural and artificial Systems*, volume 5th edition printed in 1998. MIT Press.

International Organization for Standardization (2006). *Application Protocol: Mechanical product definition for process planning using machining feature*. Industrial automation systems and integration - Product data representation and exchange, Part 224. Beuth, Geneva, 3rd edition.

Jong, K. D. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. University of Michigan.

Loukil, T., Teghem, J., and Tuyttens, D. (2005). *Solving multiobjective production scheduling problems using metaheuristics.*, volume 161 of *European Journal of Operational Research*. Elsevier Ltd.

Michalewicz, Z. (1999). *Genetic algorithms + data structures = evolution programs*. Springer Verlag.

Nissen, V. (1997). *Einfhrung in Evolutionre Algorithmen*. Vieweg.

Syswerda, G. (1989). *Uniform crossover in genetic algorithms*. Proceedings of the third international conference on Genetic algorithms. Morgan Kaufmann Publishers Inc.

Syswerda, G. (1991). *Schedule optimization using genetic algorithms*. Handbook of genetic algorithms.

Ting, C.-K., Su, C.-H., and Lee, C.-N. (2010). *Multi-parent extension of partially mapped crossover for combinatorial optimization problems*, volume 37 of *Expert Systems with Applications*. Elsevier Ltd.

T'kindt, V. and Billaut, J.-C. (2006). *Multicriteria Scheduling - Theory, Models and Algorithms*, volume 2nd. Springer Verlag.

Unversity of Tuebingen. http://www.ra.cs.uni-tuebingen.de/software/EvA2/ - accessed on 29.01.2011.

Vnyi, R. (2004). *Object oriented design and implementation of a general evolutionary algorithm*. Conference for Genetic and Evolutionary Computation - GECCO 2004.

Weicker, K. (2007). *Evolutionre Algorithmen*, volume 2. Teubner Verlag.